A Dynamic Approach to MIB Polling for Software Defined Monitoring

A Dynamic Approach to MIB Polling for Software Defined Monitoring

*Md Israfil Biswas, *Mamun Abu-Tair, *Philip Morrow, *Sally McClean, *Bryan Scotney and **Gerard Parr
*School of Computing and Information Engineering, Ulster University, Northern Ireland, UK
**School of Computing Sciences, University of East Anglia, UK
Email: *{mi.biswas, m.abu-tair, pj.morrow, si.mcclean, bw.scotney}@ulster.ac.uk, **g.parr@uea.ac.uk

***Abstract*—Technology trends such as Software-Defined Networking (SDN) [1][2][3] are transforming networking services in terms of flexibility and faster deployment times. However, SDN-like services still require network management to ensure security and efficiency of the network. A traditional Network management system (NMS) usually adopts the Simple Network Management Protocol (SNMP) [4] as its management protocol and has achieved great success. However, with emerging services, traditional network management has some shortcomings. Software-Defined Network (SDN) divides the control plane from the data plane in contrast to the distributed control of current networks. The advanced model can simplify the complexity of the traditional network management with a centralized architecture. However, SDN still lacks efficient management tools and is difficult to realise in a full deployment, which makes network operators reluctant to completely replace their legacy NMS. Hence, many studies have been done to merge SDN with traditional NMS. Results show that merging SDN with traditional NMS not only increases the average Management Information Base (MIB) polling time but also creates additional overheads on the network. Therefore, this paper proposes a dynamic scheme for MIB polling using an additional MIB controller agent within the SDN controller. Our results show that using the proposed scheme the average polling time can be significantly reduced (i.e., faster polling of the MIB information) and also it requires very low overhead because of the small sized OpenFlow messages used during polling.**

Keywords— SDN, MIB, OpenFlow, Monitoring, SNMP

I. INTRODUCTION

With the popularity of cloud computing features e.g., network virtualisation [5][6], live migration [7][8] etc. various network services are deployed on the Internet for dynamic resource provisioning. However, these advances generate a considerable amount of Internet traffic volume and required advance network management for security and high efficiency. The conventional network devices are designed and configured for basic Internet access services, and therefore are static and inflexible in their physical hardware implementation. As a result, existing networking devices required frequent updates as new network services are continuously deployed.

Network management includes various accountabilities e.g., operation, administration, maintenance, and provisioning of network systems, which are very important for networks. With the rapid growth of networks new security, efficiency or other problems pose threats to large Datacentres (DCs) or enterprises. The analysis of network utilization is the key for a network manager. Therefore, traffic monitoring is common and significant in existing NMSs. An efficient NMS can monitor the network effectively and usually provides network utilization information per trunk or link. NMS can offer a more rapid, accurate network view to ensure and optimize the network operation. Traditional NMS usually adopts SNMP as its network management protocol which is the most widespread protocol in network management for exchanging management information between network devices. SNMP is popular because of vendor support and is based on a manager-agent model to provide a unified data communication. SNMP Agents run on network equipment and all the management information is stored in a MIB. NMS can get or set parameters of equipment through SNMP.

Although the traditional network management has achieved great success, with the emergence of various kinds of networking services such SDN, the traditional management system exposes inadequacies when deployed in DCs and enterprises with new architectures. This results in low data forwarding efficiency, complex network management, and shortage of network addresses. Hence, to satisfy both users and network operators, it is necessary to find a flexible, intelligent and robust network management approach.

Network virtualization techniques allow service providers to slice infrastructure resources, enabling a flexible deployment of new network technologies. SDN breaks the traditional hardware barrier by introducing reconfigurable and extensible modules in network devices by separating the control plane from the data plane. SDN increases network flexibility and service agility with resource provisioning. Hence, continuous monitoring of SDN traffic is also required for utilisation of the network resources. SDN manages data flows and switching using the OpenFlow protocol [9] whereas, SNMP has been widely used in TCP/IP-based networks for the monitoring of network elements and hosts. However, the monitored devices are represented as managed objects and defined as a MIB. Network traffic statistics via SNMP corresponds to periodic polling of MIB objects (for example, *ifTable* objects in *MIB II*). Hence, periodic MIB polling is required for continues monitoring.

SNMP largely deals with the management plane where focus is on collecting information about the traffic and status of the elements and is typically consumed by a NMS through polling the information periodically. Hence, the management plane monitors and configures the network element. Whereas, the control plane defines how packets flow through the network element. OpenFlow, by definition focuses on the Control Plane but also supports the Management Plane of the network.

OpenFlow-like protocols are required to implement the SDN paradigm using the new network elements to incorporate Network Function Virtualization [10]. Real-time higher level executable policies across the management control plane between the main network elements are also required to expose underlying performance attributes across the end-to-end system. Hence, new enterprise MIB schema is required for agile cloud enterprise MIB data structure.

The work described in this paper is conducted as part of a wider US-Ireland funded project concerned with enabling efficient and secure cloud computing for high capacity applications, including dynamic optical Terabit scale networking. Software Defined Monitoring with a MIB will necessitate real-time higher level executable policies across the management control plane between the main network elements. The MIB schema and syntax together with a policy engine will be capable of allowing the SDN controller to make real-time decisions about the cost and benefits of migration and/or replication. In particular, in this paper we initially used the SNMP protocol for MIB polling through the SDN controller i.e., merging the SDN controller with traditional NMS and found that using SNMP not only increases the average MIB polling time but also creates significant overheads on the network. Hence, this paper proposes a dynamic scheme of MIB polling using an additional controller inside the SDN controller. Our results show faster polling of the MIB information and very low overhead in the network compared to the NMS MIB polling.

The rest of the paper is organised as follows: Section II describes the related work in this area and how our work is unique, Section III provides details of the Software Defined Monitoring techniques and illustrates our proposed scheme. Section IV describes the experimental setup and configuration for this work. Section V presents the results with discussions comparing the traditional NMS with SDN and our proposed scheme. Finally, section VI provides some conclusions and a view for future work.

II. RELATED WORK

Many research studies related to network management have been undertaken. However, new network architectures with NMS required SNMP-like traditional network management protocols to manage the architecture effectively. In [11], a SNMP based model CNMM has been developed for cloud networks. The proposed model provides a solution to manage the growing traffic in the cloud and improve communication of manager and agents as in SNMP.

A management architecture and Manager-agent communication model has been modelled in [12] to coordinate the information residing on the single elements of the multi-stage router. The model presents a unified view to the external network management station concerning requests from SNMP.

Much research also has been conducted without using the traditional SNMP for Software Defined Monitoring. John et al proposed a split selected monitoring control functionality onto node-local control planes in [13]. It takes the advantage of processing capabilities on programmable nodes. Their approach is a rate monitoring function in SDN that is implemented using node-local control plane components introducing a messaging bus for simple and flexible communication between monitoring function components as well as control and management systems. They claim that their rate monitoring approach generates only a tiny fraction of the monitoring traffic from comparable SNMP and OpenFlow implementations, while providing the same information granularity.

However, the main issue here is in considering the entire infrastructure as one unified service production environment and therefore, the challenge is to provide up-to-date, accurate, and detailed monitoring information to orchestration and control layers in a scalable way.

In optical networks, Non-SDN Reconfigurable Optical Add-Drop Multiplexer (ROADM)s are not always able to update hardware or software in the ROADMs to adapt legacy SDN architecture. In [14], a software defined monitoring architecture

has been deployed using SNMP protocol for Optical Non-SDN ROADMs. They have proposed an architecture using a proxy that translates OpenFlow messages sent by Open Network Operating System(ONOS) into SNMP messages to configure the ROADMs. The solution is for flexible monitor and manage an optical network via SDN architecture.They claim that their solution is also able to recover and reroute wavelengths when a link is down. The adapted solution to legacy networks does not require any upgrade on the optical network elements. The proposed SDN architecture is adapted to include legacy non-SDN ROADMS.

Although they claim their proposal does not require any software modification of the SDN controller or ROADM SNMP agent, the issue in such architectures is that it uses a proxy that translates the OF messages sent by the controller into SNMP commands to apply the desired configurations on the ROADM and vice-versa. This can increase delay in a large datacentre or inter-datacentre networks and reduce the monitoring performance.

In [15], an efficient scheme for performance management is developed to collect traffic statistics data via the SDN controller plane. The scheme proposes a periodic collection and transfer of MIB objects for bulk traffic statistics collection. The scheme is developed in the controller plane and provides a northbound interface for upper network management applications. Instead of using SNMP and MIBs, the scheme is implemented by periodically gathering statistics information of flow tables from SDN-enabled switches via the OpenFlow protocol. However, the issues here are the various OpenFlow packet sizes that creates overheads in the network. The architecture also considers possible performance degradation in the SDN controller for additional controllers and distributed task queues to achieve high availability and scalability. However, our work is motivated for such architectures and considered very small packets for SDN monitoring. Therefore, our scheme not only reduces the overall network overhead but also achieves high speed data polling.

In summary, this paper is unique in the following aspects:

- This work uses small packet sizes i.e., only 64-byte OpenFlow packets for SDN monitoring and hence can perform high speed polling.
- Small sized packets will also reduce the overall network overhead for SDN monitoring techniques and therefore can improve the QoS of the datacentre.
- The architecture achieves high availability by ensuring reduced latency between the SDN controller and the developed additional MIB controller with scalable efficient task queues.

The next Section describes traditional network management with MIB using SNMP protocol and proposes a dynamic approach of MIB polling in a software defined network for centralised network monitoring.

III. SOFTWARE DEFINED MONITORING

This paper aims to develop a dynamic approach for MIB polling in SDN for monitoring. Our proposed approach includes an additional MIB controller agent in the controller plane of SDN. The MIB controller agent is designed considering a loosely coupled architecture for MIB polling to support high availability and scalability as defined in OpenFlow 1.2 or later.

A. Management Information Base (MIB)

SNMP agents (e.g., Net-SNMP) are allowed to collect the management information database from the device locally and make it available to the SNMP manager. Hence, the agent maintains an information database describing the managed device parameters.

NMS uses this database for specific information and this commonly shared database between the Agent and the Manager is called a MIB. A MIB is basically a collection of information for managing network elements. The MIB contains a standard set of statistical and control values defined for hardware nodes on a network. Private MIBs extends these standard values with values specific to a particular agent.

The MIBs contains of managed objects identified by the name Object Identifier (Object ID or OID). Each Identifier is exclusive and represents specific features of a managed device. However, the return value of each identifier could be different e.g. Text, Number, Counter, etc. Like a folder structure on PCs, OIDs are very structured, and follow a hierarchical tree pattern as shown in Fig.1. However, unlike folders all SNMP objects are numbered. Therefore, the top level is the root and after the root is ISO with the number “1”. ORG is the next level with the number “3” as it is the 3rd object under ISO. OIDs are always written in a numerical form, instead of a text form.

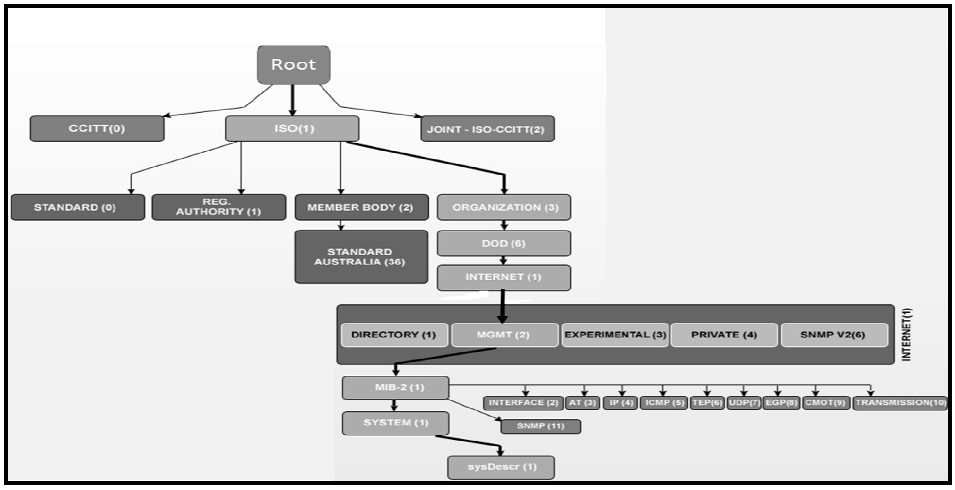

Fig. 1. MIB The Registered Tree

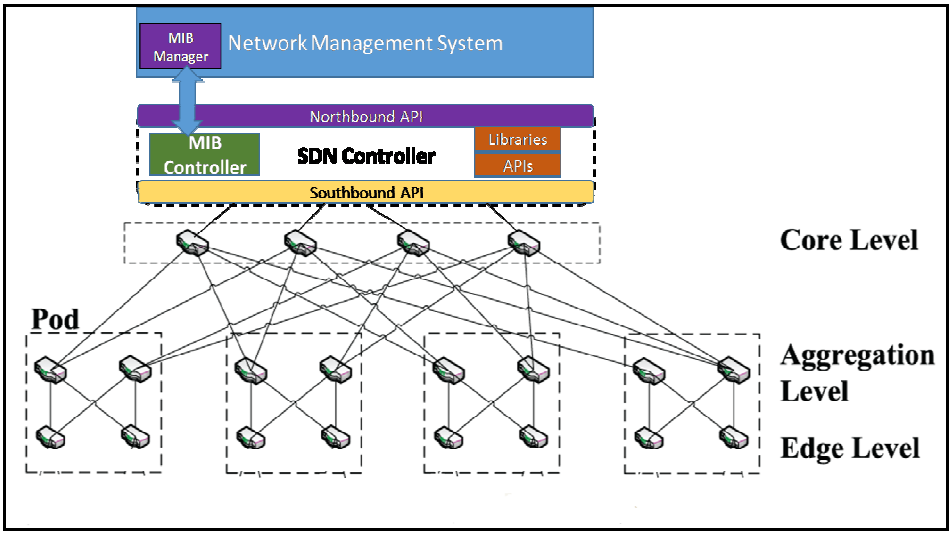

Fig. 2. MIB Polling Scheme with NMS initiated in K-ary Fat Tree Topology

For example, three object levels are written as 1.3.0 not iso\org\standard. As shown in the figure, a typical object ID will be a dotted list of integers. Hence, the OID in RFC1213 for sysDescr is .1.3.6.1.2.1.1.1 and using the OID the system can get the hardware and software information used on the host.

B. NMS with SNMP

In NMS, SNMP polls MIB information and gets a response from its MIB agents (e.g., switches, routers). Fig.2 shows a network management system that polls information by sending a request through the SNMP Manager and gets a response from a SNMP agent. An agent can send a spontaneous TRAP to the NMS if required. SNMP TRAPs are initiated by the agents and the agent sends the TRAP to the SNMP Manager on the occurrence of an event.

NMS using SNMP fetches MIB information directly from network devices for traffic monitoring. The collection of managed object values is performed periodically and then the information can be automatically transferred to a database. Under the NMS control via SNMP protocol, polling is still a popular mechanism to gather information from the managed networks. Most NMSs collect data from network elements directly via SNMP. However, in recent developments of datacentre networks, OpenFlow based SDN requires monitoring of network devices and there has not yet been sufficient research done on SDN monitoring.

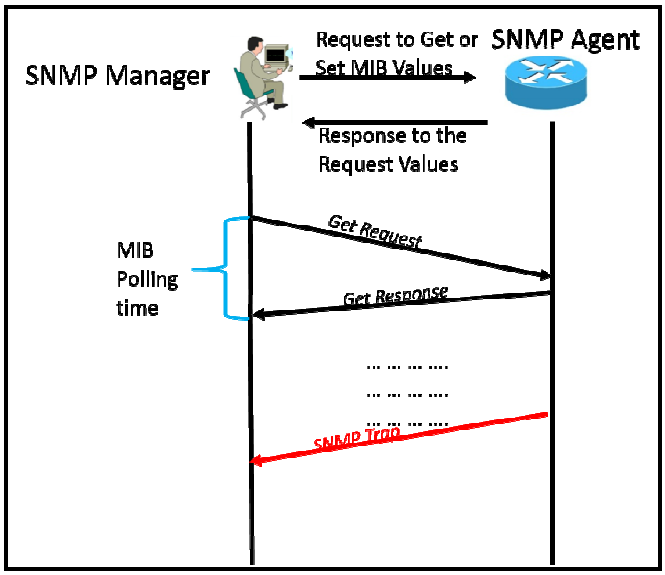

Fig. 3. Illustration of MIB Polling

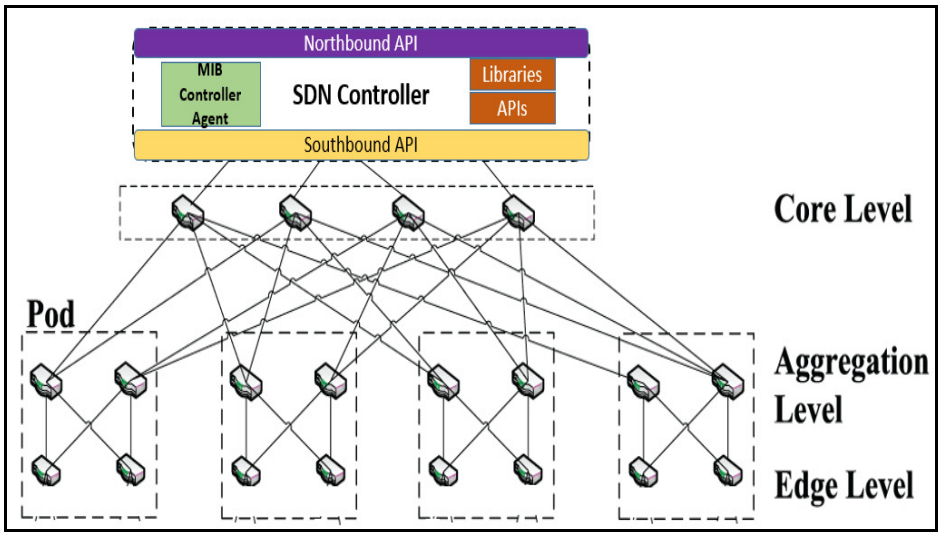

Fig. 4. MIB Polling Scheme with Proposed Approach in K-ary Fat Tree Topology

C. NMS with SDN

This work first aims to develop a MIB polling mechanism for SDN monitoring through the NMS using SNMP. As shown in Fig.3, we have introduced a MIB manager at the NMS to bring a change in management paradigm from a distributed traditional NMS to a centralized SDN control. The MIB manager fetches MIB information from the SDN controller. The manager provided in our NMS is to get MIB information through the management plane service over the SNMP protocol. The MIB data are delivered in the SDN when requested. Therefore, NMS can easily access MIB data for monitoring using SDN controller as supported in OpenFlow.

D. The SDN MIB Controller Agent

SNMP was envisioned for exposing data to external applications for remote monitoring. A distinctive feature of SNMP includes the capability of sending trap messages so that the agent device can push information about their status or condition to the management plane. However, SNMP has many shortcomings, including being limited in the number of data types it can handle. The vendors can extend the SNMP OID in their own numbering scheme but the extension does not solve the whole problem with the advances of the emerging technologies like SDN.

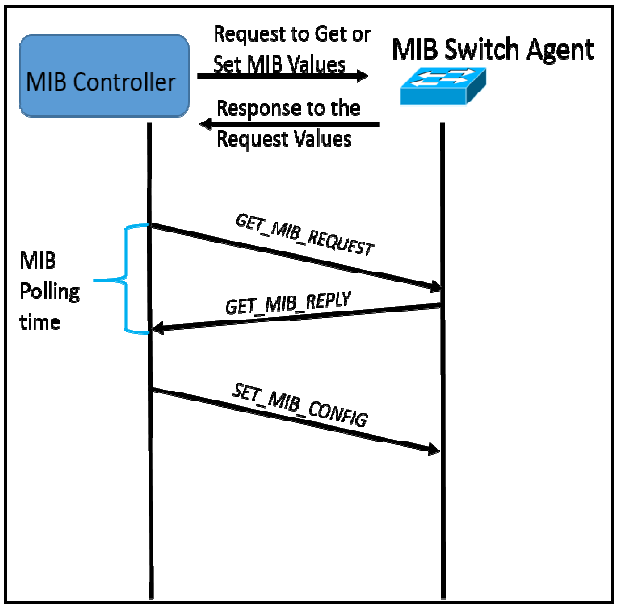

Fig. 5. Illustration of MIB Polling in SDN environment

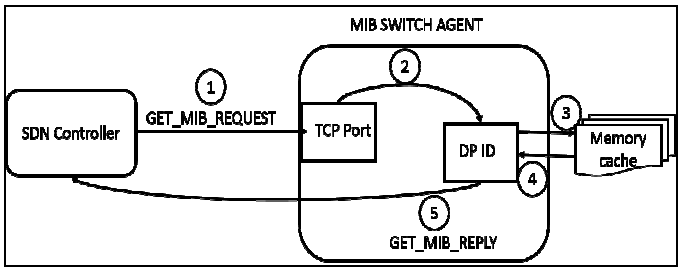

Fig. 6. State Diagram of MIB Polling in SDN environment

Hence, in this paper we have introduced a MIB controller agent in the SDN controller using the RYU SDN Framework development [16] as shown in Fig.4. The MIB controller agent can set and query MIB configuration parameters in the switch with the SET_MIB_CONFIG and GET_MIB_REQUEST messages. The switch responds to a MIB value request with an GET_MIB_REPLY message. Moreover, like the OpenFlow switch reply messages, it does not reply to a request to set the configuration as shown in Fig.5.

The MIB controller agent in the SDN controller is implemented as a controller agent which sends MIB requests to a TCP port by using Netcat [17] to generate Traffic in a Mininet topology. In this work, for simplicity we have stored the exact MIB information in the Switch agent memory cache as we have used for the SNMP MIB information. Fig.6 shows the state diagram used for MIB polling:

- In *Step 1*: The *GET_MIB_REQUEST* is sent by the controller, which is a small TCP Packet using Netcat. We have used 64-byte frames to generate high packet rates and force high packet processing in the OpenFlow switch from the MIB controller.
- In *Step 2*: With the help of Wireshark, we are able to trace corresponding Data-path IDs (DPID) of the OpenFlow switch and we have maintained a TCP Port to DPID table for this experiment to dynamically forward the MIB request to the memory cache.
- In *Step 3*: The DPID finally requests the MIB information from the Memory cache.
- In *Step 4*: The switch returns the MIB info as OpenFlow small 64-byte packets to the DPID.
- In *Step 5*: The info is return as the *GET_MIB_REPLY* to the SDN controller directly.

For better performance, the MIB information is written in an in-memory cache maintaining a single list. The MIB data then can be dynamically configured via the northbound interface of the controller for monitoring. We have used the *miss_send_len* field in the OpenFlow that defines the number of bytes of each packet sent to the controller to reduce the packet size to

generate high packet rates and force high packet processing in the OpenFlow switch from the MIB controller [18]. The *miss_send_len* is set to 64-bytes for small packets, whereas the default is flexible in OpenFlow version 1.3. The *ofctl_v1_3* sends 0-byte length data in a *packet_in* message if *max_len* is not specified, which is 65535.

In NMS, SNMP allows PDUs sized up to the MTU of the network i.e., Ethernet allows up to 1500-byte frame payloads [19]. Therefore, in each MIB polling, our proposed approach can reduce noticeable network overhead. Moreover, in each polling interval, we can reduce overhead of (16 X 2872=45952byte) or 45.9kB from 16 active MIB switch agents at one polling compared to the NMS MIB polling approach (for a general calculation, here we are not considering the retransmitted packets)..

IV. EXPERIMENTS AND RESULTS

A. *Setup and Configuration*

We have used Mininet version 2.2.1 and OpenFlow version 13 running on an Intel(R) Core(TM) i7 3.40 GHz CPU with 16GB of memory for the experiments. All the experiments are done over 1000 runs with 0.95 or, 95% confidence interval [20]. All the polling times in this paper are measured using Wireshark traces [21]. Table I shows the configuration details for the fat tree topology.

We started some initial capacity variation experiments using SNMP and the SDN controller in a fat tree topology to check the Mininet topology. SDN commenced with SNMP protocol shows that the average polling time is lower with respect to the higher link capacities between the Top of Rack and the Aggregate level switches. We found the gigabit links takes only a few milliseconds for MIB polling on average whereas, the average time for MIB polling can be up to 50 times higher using 100Mbps links compared to the gigabit links as expected. We have performed a number of experiments described in various scenarios; the next sub-sections present the experimental results:

- **The first scenario** considers a comparison between the developed MIB Manager in the NMS Application with the proposed additional MIB controller agent at the SDN without background traffic.
- **The second scenario** continues the comparison considering various amounts of background traffic.

TABLE I. CONFIGURATION OF K-ARY FAT TREE TOPOLOGY SCENARIO 1

Parameters	**Value**
Servers/Hosts	16
Top of Rack switches	8
Aggregate switches	8
Core switches	4
Bandwidth	
Host to Top of Rack (EDGE Level) switches	1Gbps
Aggregate to Top of Rack (EDGE Level) switches	1Gbps
Aggregate to Core switches	1Gbps
Polling Interval	60 sec

Test Scenario 1

The first scenario is chosen to measure the polling speed considering a datacentre with no background traffic. We have compared the Average Polling time for the developed MIB Manager in the NMS with the proposed MIB controller agent in the SDN by varying the MIB switch agents. Using NMS, the MIB Manager gets the bandwidth of the interface to the MIB switch agent. The *IfSpeed* variable is used in this case that replies with the speed of the interface as reported in the SNMP *ifSpeed* object. Our proposed approach requests similar MIB information that has been stored in the MIB switch memory cache considering the fat tree topology using Mininet.

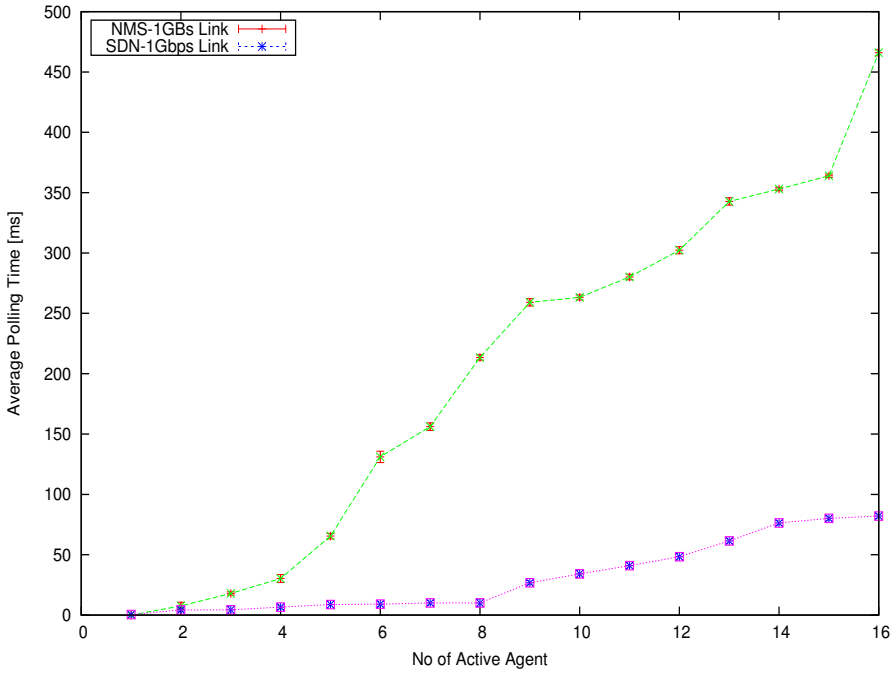

Fig. 7. Average polling Time [ms] with No Backgound traffic

Fig.7 shows that the Average Polling time initiated by the MIB controller can be up to nine times lower compared to the polling initiated by NMS. The reason is, when the NMS required any polling, it uses the MIB manager at the application level and send the request via the SDN controller. The SDN controller checks its port information and forwards the request to the associated MIB agent. Hence, this requires a number of stages to send the request to the MIB switch agent and certainly add delays. In contrast, our approach directly sends MIB request to the switch and the switch fetches the MIB info from the Memory cache and returns back directly to the SDN controller.

The figure also shows that with the number of increased active MIB switch agents, the average polling time difference between the two approaches increases.

For example, while 4 MIB switch agents are active, the average polling time initiated by NMS is 26ms, whereas our approach shows only 7ms. However, with 16 Active Switch Agents the polling time initiated by NMS is 460ms, whereas the proposed approach can take up to 96ms.

Test Scenario 2

In the second scenario, we have considered various amount of background traffic while MIB polling to observe the overall network impact. Table II shows the configuration details used in this scenario in the fat tree topology. With various amounts of background traffic, many polling request packets are sent but didn't get a response within the keep alive time and therefore are retransmitted by the NMS. Our observation is the number of retransmissions significantly increases with the increase in background traffic during the MIB polling initiated by the NMS. We have used *iperf* [22] with UDP packets in Mininet to create background traffic flows.

With 20% background traffic, Fig.8 shows that high retransmission happens due to NMS application delays during MIB polling, i.e., packets are lost and no reply before the keep alive times. For example, with 4 active Switch agent, similar average polling times are observed by using both NMS and proposed approach, which is less than 1sec. However, the average polling time can be very high i.e., up to several minutes, whereas our MIB controller agent does not require any MIB Manager from the application plane and anticipates that latency can be shorten and polling time is minimized as shown in the figure With 16 Active Switch Agents, the figure shows that the average polling time can be up to 36sec, whereas the proposed approach shows that the average polling time can be few seconds.

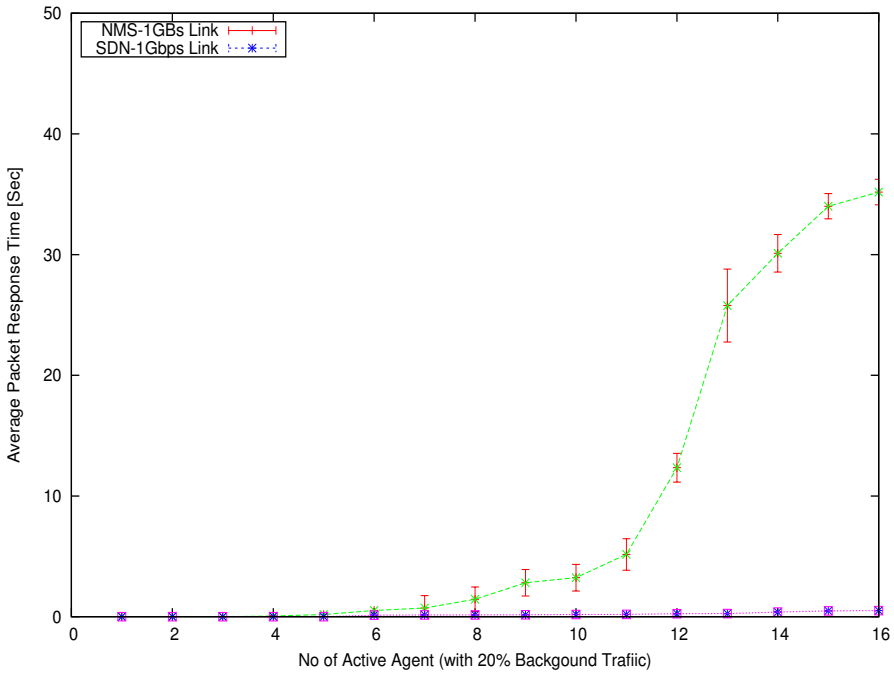

Fig. 8. Average polling Time [Sec] with Backgound traffic (20%)

TABLE II. CONFIGURATION OF K-ARY FAT TREE TOPOLOGY:SCENARIO 2

Parameters	**Value**
Servers/Hosts	16
Top of Rack switches	8
Aggregate switches	8
Core switches	4
Bandwidth	
Host to Top of Rack (EDGE Level) switches	1Gbps
Aggregate to Top of Rack (EDGE Level) switches	1Gbps
Aggregate to Core switches	1Gbps
Background Traffic	UDP
Background Traffic with respect to Network bandwidth	20%, 50% and 80%
Polling Interval	60 sec

The impact of the retransmissions can be observed in Fig. 9, the overall packet drop was less than 1% when the number of Active MIB Switch is 4 using both approaches and it has increased up to 11% when the number of switches has increased to 16 using the NMS MIB polling approach. However, using the proposed approach the average packet drops observed is 2% for 16 MIB switch agents.

We have also obtained full sets of results considering 50% and 80% background traffic. Fig.10 shows that considering 50% background traffic on the link, more requested MIB packets are lost or, the MIB info reply has not arrived within the keep alive time compare to the 20% background traffic. For example, the figure shows that the average response time is less than a second while the number of active switch agents is 3 using both approaches and it can increase up to 56sec using NMS MIB polling. However, using the proposed approach it increases only up to 5sec.

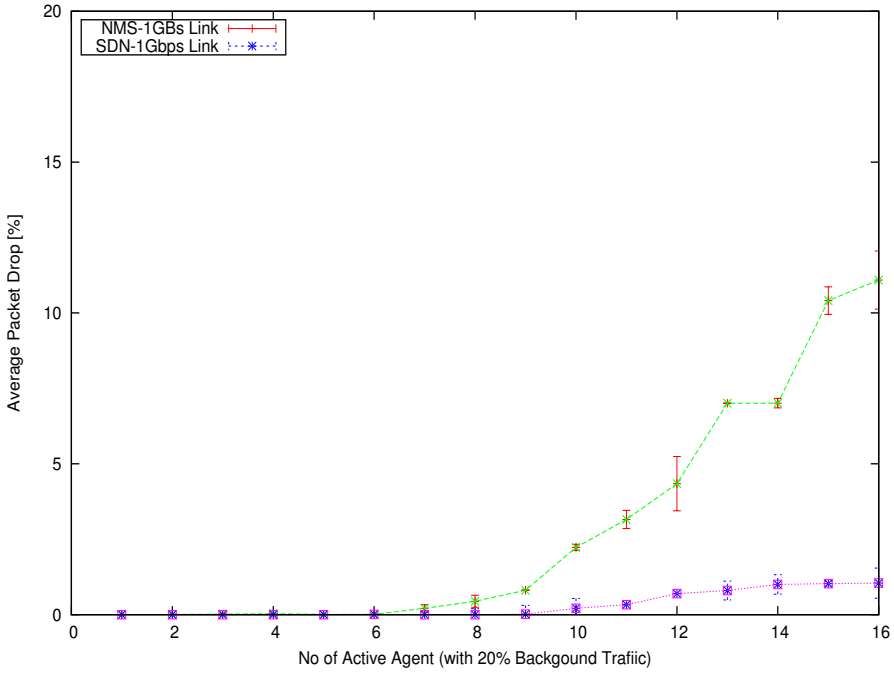

Fig. 9. Average packet Drops with Backgourd Traffic (20%)

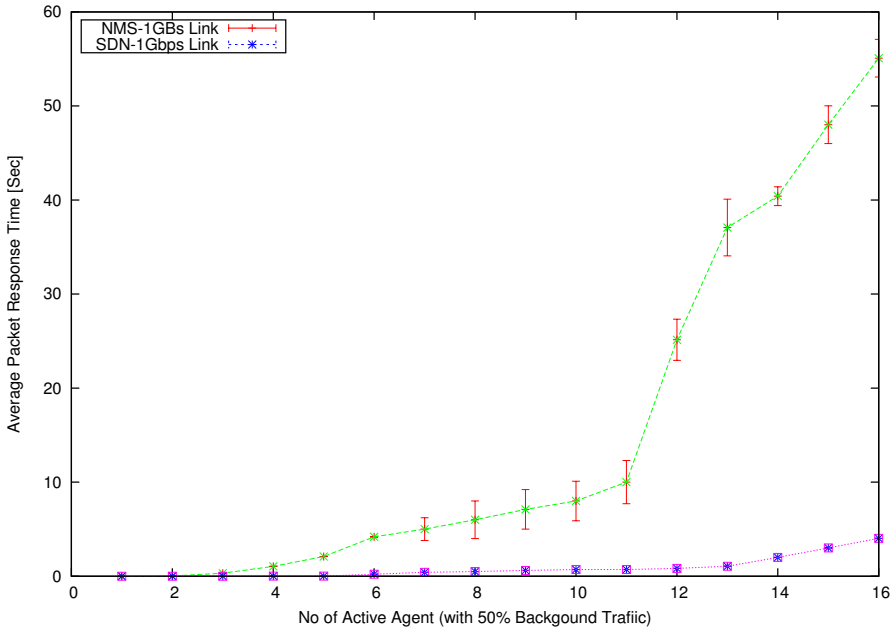

Fig. 10. Average polling Time [Sec] with Backgourd Traffic (50%)

The packet drops graph considering 50% background traffic also shows that the average packet drops also increases compared to 20% background traffic as shown in Fig.11. It can be up to 22% for 16 active switches using NMS MIB polling, whereas using our approach the packet drops increased to 4%.

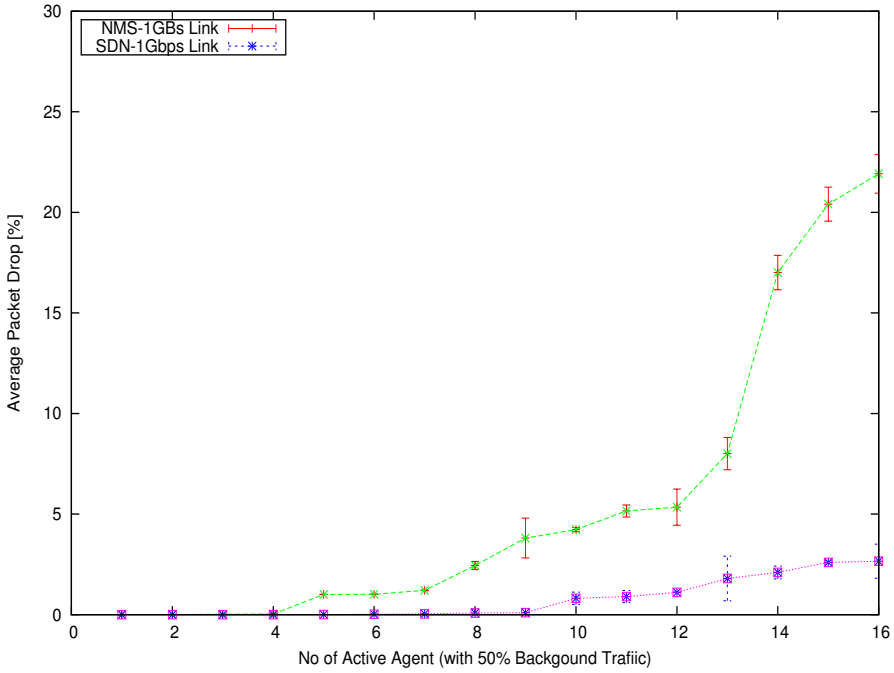

Fig. 11. Average packet Drops with Backgourd Traffic (50%)

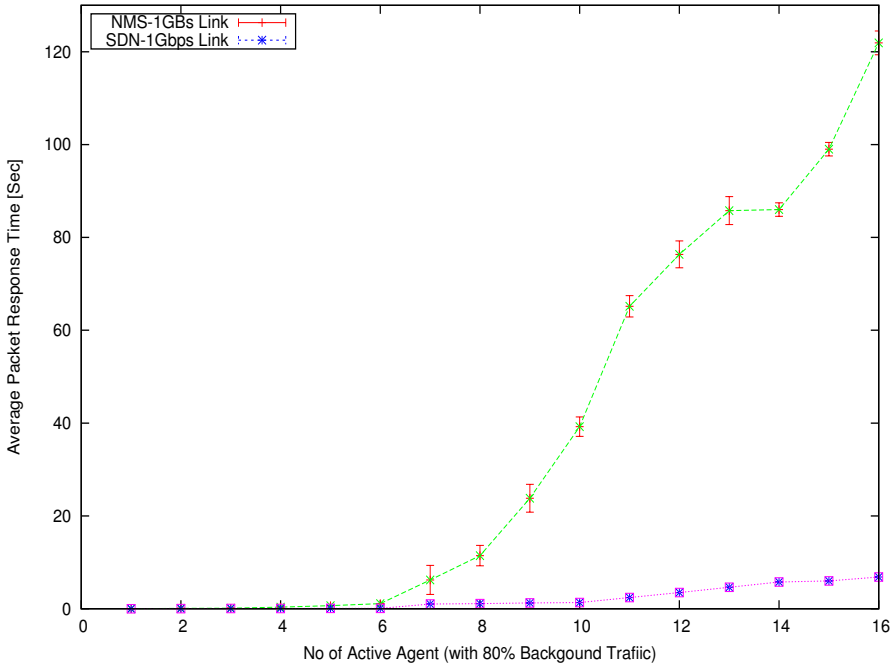

Fig. 12. Average polling Time [Sec] with Backgound traffic (80%)

Considering 80% background traffic, the average polling time can be very high i.e., up to several minutes using NMS MIB polling, whereas our MIB controller agent does not required any MIB manager from the application plane and it is anticipated that latency can be shorten and polling time is minimized as shown in Fig.12. For example, considering MIB polling using NMS, many retransmissions occur while 16 active switch agents are replying to the MIB requests and the average polling time observed is up to 118sec. However, our approach shows that the average polling time is less than 6 sec.

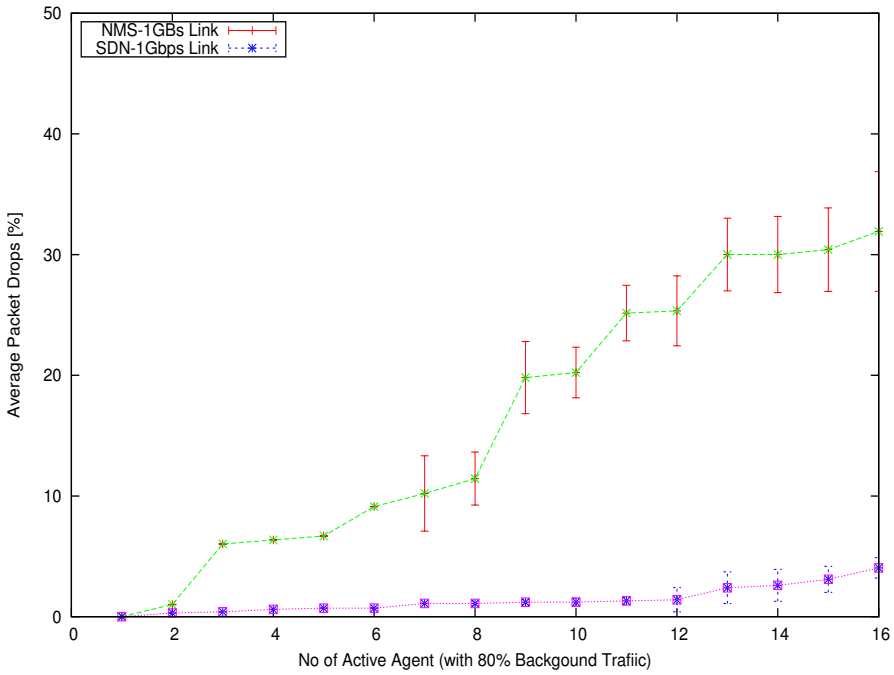

Fig. 13. Average packet Drops with Backgourd Traffic (80%)

The effect of the increased retransmissions can be observed in Fig. 13, where the overall packet drops can be up to 40% using the NMS MIB polling approach when the number of active MIB switch agents is 16. However, using the MIB controller as proposed in this paper, the maximum overall packet drops can be around 6%.

V. Conclusion and Future Work

Network monitoring is essential for network management where MIB polling from network devices is well recognised. Traffic monitoring using MIBs help network operators understand network traffic volume and bandwidth utilization, and are also important for network planning and design. In this paper, we have proposed a dynamic approach to effectively collect MIB information for SDN, and implemented the proposed architecture with an SDN controller to confirm its feasibility. Furthermore, we addressed issues in the MIB polling initiated by the NMS via SDN and proposed effective solutions.

In SDN, a flow could be related to Inter-DC or Intra-DC. Accordingly, it is possible to attain more detailed MIB traffic in SDN, for example, network traffic consumed by an optical network or application. The low level optical attributes can be augmented with a formal illustration of the current network configuration and traffic load which will be closely coupled to the scheduling algorithms that will suggest reconfigurations to the SDN controller to be pushed down to the network elements. This formal representation of the network can monitor data from the network to be maintained on a per-link basis: average queuing delay, data loss, modulation scheme, encoding scheme, throughput, utilization, jitter and other metrics that will become available from fast optical switching. Hence, the proposed scheme is useful for many network management applications. However, sending small packets could result in lower throughput and therefore a network administrator's choice to trade-off between the throughput and the polling response time in the case of high speed polling for network monitoring without interfering with the network data traffic. Future work will further investigate and develop high speed polling mechanisms considering real datacentre environments.

Acknowledgements

This research is supported by the 'Agile Cloud Service Delivery Using Integrated Photonics Networking' project funded under the US-Ireland Programme (NSF (US), SFI (Ireland) and DEL (N. Ireland).

REFERENCES

[1] Feamster, N., Rexford, J., and E. Zegura, "The Road to SDN: An Intellectual History of Programmable Networks", ACM Queue, Volume 11, Issue 12, 2013.

[2] Kreutz, D., Ramos, F., Verissimo, P., Rothenberg, C., Azodolmolky, S., and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey", Networking and Internet Architecture (cs.NI), arXiv:1406.0440, 2014.

[3] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <http://www.rfc-editor.org/info/rfc7426>.

[4] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol", STD 15, RFC 1157, SNMP Research, Performance Systems International, MIT Laboratory for Computer Science, May 1990.

[5] Chowdhury, N. and R. Boutaba, "Network Virtualization: State of the Art and Research Challenges", Communications Magazine, IEEE, Volume 47, Issue 7, pp. 20-26, 2009.

[6] M. I. Biswas, G. Parr, S. McClean, P. Morrow and B. Scotney, "SLA-Based Scheduling of Applications for Geographically Secluded Clouds", 1st workshop on Smart Cloud Networks & Systems (SCNS'14), December 3-5 Paris, France, 2014.

[7] M. I. Biswas, G. Parr, S. McClean, P. Morrow and B. Scotney, " A Practical Evaluation in Openstack Live Migration of VMs using 10Gb/s Interfaces", The Second International Workshop on Education in the Cloud-EC2016, 29 March-2 April 2016, Oxford, UK.

[8] M. I. Biswas, G. Parr, S. McClean, P. Morrow and B. Scotney, "An Analysis of Live Migration in Openstack using High Speed Optical Network", IEEE Technically Sponsored SAI Computing Conference, 13-15 July, 2016, London, UK.

[9] ONF, "OpenFlow Switch Specification Version 1.5.0," https://www.opennetworking.org/images/stories/downloads/sdnresources/onf-specifications/openflow/openflow-switchv1.5.0.noipr.pdf, December 2014.

[10] Network Functions Virtualisation (NFV); Infrastructure; Hypervisor Domain, ETSI GS NFV-INF 004 V1.1.1 (2015-01), http://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/004/01.01.01_60/gs_nfv-inf004v010101p.pdf

[11] M. Madan and M. Mathur, "Cloud Network Management Model A Novel Approach to Manage Cloud Traffic," International Journal on Cloud Computing: Services and Architecture (IJCCSA),2014.

[12] A. Bianco, R. Birke, F. G. Debele and L. Giraudo, "SNMP Management in a Distributed Software Router Architecture," 2011 IEEE International Conference Communications(ICC), pp. 1-5, June, 2011

[13] W. John, C. Meirosu, B. Pechenot, P. Sköldström, P. Kreuger and R. Steinert, "Scalable Software Defined Monitoring for Service Provider DevOps", Proc. EWSDN 2015, Bilbao, Spain, October 2015.

[14] Imad Alawe, Bernard Cousin, Olivier Thorey, Rodolphe Legouable, "Integration of Legacy Non-SDN Optical ROADMs in a Software Defined Network", , vol. 00, no. , pp. 60-64, 2016, doi:10.1109/IC2EW.2016.11

[15] T.Wang, Y. Chen, S. Huang, C. Hsu, B. Liao and H. Young, "An efficient scheme of bulk traffic statistics collection for software-defined networks", APNOMS 2015, pages 360-363.

[16] Ryu SDN Framework, https://osrg.github.io/ryu/

[17] Netcat: the TCP/IP swiss army, http://nc110.sourceforge.net/

[18] Otto Carlos M. B. Duarte and Guy Pujolle, "Virtual Networks: Pluralistic Approach for the Next Generation of Internet ,Technology & Engineering,Wiley | Wiley-ISTE, Aug 12, 2013.

[19] R. Presuhn, J. Case, K. McCloghrie, M. Rose and S. Waldbusser, "Protocol Operations for SNMP",RFC 3416,December 2002.

[20] Danny Yuxing Huang, Kenneth Yocum, Alex C Snoeren, "High-fidelity switch models for software-defined network emulation", Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking pp.43-48, Hong Kong, China , August, 2013.

[21] Wireshark User's Guide, https://www.wireshark.org/download/docs/user-guide-a4.pdf

[22] iPerf - The network bandwidth measurement tool, https://iperf.fr/